IN THE SPECIFICATION

Please amend Paragraph [0008] as follows:

[0008] In one embodiment of the naming service, the binder module is operable to associate the name of a [[java]] JAVA service object to the location of a [[java]] JAVA naming and directory interface having a reference to the [[java]] JAVA service object. A name look-up module is operable to provide a requesting application with the remote [[java]] JAVA naming and directory service. The location information, for example may contain the provider url, initial context factory, the associated [[java]] JAVA naming and directory interface and/or the full home interface name.

Please amend Paragraph [0010] as follows:

[0010] In one embodiment an enterprise naming service for applications to locate services is provided. The enterprise naming service for applications to locate services comprises a binding module to associate an interface maintaining a reference to a first service with a location of the interface, the binding module further operable to associate a second service with a location of the second service and a look-up module to provide the location of the interface in response to a request by an application, the look-up module further operable to provide the location of the second service in response to a request by a second application. For example, in one embodiment of the naming service for applications to locate services, the binder module associates a [[java]] JAVA naming and directory interface service maintaining an address or universal reference locator of an enterprise [[java]] JAVA bean with the address or universal reference locator of the [[java]] JAVA naming and directory interface service, and the look-up module provides the address or universal reference locator of the [[java]] JAVA naming and

directory interface service in response to a request by an application for information on how to access the enterprise [[java]] JAVA bean. In another example, in one embodiment the binder module is operable to associate the name of a CORBA object to an address or reference to the CORBA object, and the name look-up module is operable to use the association to provide the address or reference of the CORBA object to an application which has requested the look-up information of the CORBA object. The application may then use this CORBA object address or reference to directly invoke the services of the CORBA object.

Please amend Paragraph [0019] as follows:

[0019] Figure 5B illustrates a sequence diagram of messages employed to access an enterprise [[java]] JAVA bean object through the enterprise name service system.

Please amend Paragraph [0024] as follows:

[0024] Turning now to Figure 1 a block diagram of an enterprise name service (ENS) system 10 is depicted. Service providers 12a, 12b, and 12c are computer programs or applications which provide services to client applications 14a, 14b, and 14c. In some cases the service providers 12 may be interfaces which provide a mechanism for finding service objects, for example enterprise [[java]] JAVA bean objects, which may fulfill the request of the client application 14. The ENS system 10 provides service location transparency, for example, the service providers 12 may be relocated, and the client application 14 need not change its behavior when employing the ENS system 10 to find the service objects.

Please amend Paragraph [0025] as follows:

[0025] Stated in another way, of particular relevance to the present disclosure is the potential of addressing a number of different environments or domains within an enterprise in an integrated and cost effective manner. An embodiment of the present disclosure provides a single name look-up service which provides access to name look-up for some domains directly, while for other domains provides a response to the name look up which provides a reference to the interface of a local name service for that domain. In this manner, efficiency may be gained by allowing clients throughout the enterprise to go to a single name service regardless of domain (providing the desired transparency), while at the same time reducing the need to recreate every potential name look-up for every domain in one omnibus application. One specific example of this approach is demonstrated below as object references in the CORBA domain are provided directly, while name look-ups in a JAVA domain are provided a reference to the interface of a [[java]] JAVA naming and directory service having appropriate references for the desired name look-ups in the desired JAVA domain.

Please amend Paragraph [0032] as follows:

[0032] Binding information which is provided by service providers 12 may vary substantially among different service types. Some example bindings are provided below. An enterprise [[java]] JAVA bean (EJB) binding may comprise a service name, a type, a version, a provider universal reference locator, an initial context factory, a [[java]] JAVA name and directory interface (JNDI) name, and a full home interface. An example EJB binding is:

servicename=EJBServicePlanManager

type=EJB

version=1.0

provider_url=t3://localhost:9631

initial_context_factory=weblogic.jndi.WLInitialContextFactory

jndi_name=ExamplePlanManager

full_home_interface=com.acme.enterprise.example.PlanManagerHome

Please amend Paragraph [0033] as follows:

[0033] A [[java]]JAVA_messaging service (JMS) queue connection factory (QCF) binding may comprise a service name, a type, a version, a queue manager, a hostname, a port, and a channel. An example JMS QCF binding is:

servicename=QCF_ EMR2DAV

type=JMS

version=1.0

qmgr=EMR2DAV

hostname=205.50.183.65

port=4231

channel=SYSTEM.DEF.SVRCONN

Please amend Paragraph [0035] as follows:

[0035] A [[java]] JAVA_messaging service (JMS) Topic connection factory (TCF) binding may comprise a service name, a type, a version, a queue manager, a broker queue manager, a hostname, a port, and a channel. An example JMS TCF binding is:

servicename=TCF_ EMR3DAV

type=JMS_TCF

version=1.0

qmgr=EMR3DAV

brkmgr=EMR3DAV

hostname=205.50.183.66

port=4232

channel=SYSTEM.DEF.SVRCONN

Please amend Paragraph [0042] as follows:

[0042] The ENS system 10 supports service location transparency for the client applications 14. The service provider 12 may be relocated arbitrarily, and the client application 14 need not change its behavior when employing this ENS system 10. The ENS system 10 is a unified naming service, for example applications 14 needing access to multiple service providers 12 need resort only to the single ENS system 10 to obtain the information necessary to access the service providers 12. By contrast, for example, if multiple [[java]] JAVA 2000 enterprise edition (J2EE) interfaces were to be accessed by a client application 14, the client application would need to know the universal reference locator (URL) of each [[java]] JAVA naming and directory interface (JNDI) associated with each separate J2EE interface since there is no unified J2EE repository for names at this time.

Please amend Paragraph [0045] as follows:

[0045] The communication protocol or object access mechanism employed by the client application 14 to access the services or interfaces provided by the service provider 12 determines which of several APIs supported by the ENS 16 that the client application 14 employs. CORBA objects are bound and looked-up via a CORBA COS naming interface. [[Java]] JAVA message queue (MQ) objects and [[java]] JAVA 2000 enterprise edition (J2EE) interfaces are bound and looked-up via a [[java]] JAVA ENS API. WebServices are bound and looked-up via a universal description discovery and integration (UDDI) ENS API. For example, if the client application 14 needs to access an object made accessible to it via CORBA, the client application 14 interacts with the ENS 16 via the CORBA COS naming interface. As a second example, if the client application 14 needs to access an enterprise [[java]] JAVA bean (EJB) object, the client application 14 interacts with the ENS 16 via the [[java]] JAVA ENS API.

Please amend Paragraph [0049] as follows:

[0049] A name service browser 32 supports a hypertext markup language (HTML) world wide web interface that allows access by client applications 14 or service providers 12 to obtain a variety of service status information. The Name service browser 32 is capable of displaying a CORBA web interface and a [[Java]] JAVA web interface. The CORBA web interface provides a list of bound services and allows client applications 14 or service providers 12 to further explore and look into the details, as well as to ping the status of those services. The [[Java]] JAVA web interface provides the capability to search services by service name, version, and other attributes. The name service browser 32 is a useful tool for application administrators and service providers who bind the services, as well as clients who look-up the services. For

example, the name service browser 32 provides a variety of useful functionality including the ability to be queried for available EJB services, querying if a specified service is running, identifying all or some server instances, and determining where a service is located, for example. The client application 14 or service provider 12 sends a request message to the name service browser 32, the name service browser 32 performs operations to obtain the requested information, and the name service browser 32 returns the information to the requestor.

Please amend Paragraph [0050] as follows:

[0050] Turning now to Figure 3, the ENS system 10 is depicted as a layered stack of intercommunicating modules. An API layer 34 comprises the CORBA COS naming API 36, the [[java]] JAVA ENS API 38, the UDDI ENS API 40, and the HTML interface. Service providers 12 bind and rebind their services and client applications 14 look-up services using one of these three APIs 36, 38, 40. Service providers 12 and client applications 14 may send a query to the name service browser 32 through the HTML interface. A service layer 42 comprises the binder module 18, the name look-up module 20, the service look-up module 22, and the name service browser 32. The API layer 34 invokes the services of the service layer 42 to complete the API functions or methods invoked by the service providers 12 and client applications 14. A LDAP layer 44 provides LDAP services. The service layer 42 invokes the services of the LDAP layer 44 to complete the functions or methods invoked by the API layer 34. A datastore layer 46 provides datastore services. The LDAP layer 44 interacts with the datastore layer 46 to complete the functions or methods invoked by the service layer 42.